IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant(s):		Bradley Cain	Examiner:	Vo
Serial No.	:	09/458,190	Group Art No.:	2195
Filed	:	12/09/1999		
Atty Docket	:	120-185		
Client Ref.	:	BA0409		

Title : Expediting an Operation in a Computer System

Mail Stop Appeal Brief-Patents
Commissioner for Patents
Alexandria, VA 22313-1450

APPEAL BRIEF

This Appeal Brief is hereby submitted pursuant to the Notice of Appeal filed on May 16, 2008, and responsive to the Final Office Action dated November 27, 2007.

I. Real Party in Interest

The real party in interest is Nortel Networks Limited.

II. Related Appeals and Interferences

Appellant is not aware of any appeals or interferences that are related to the present case.

III. Status of the Claims

This is an Appeal Brief from a decision dated November 27, 2007, finally rejecting all the claims currently pending in the present application. No claims have been allowed. The currently pending claims are 1, 3-6, 8-11 and 13-15.

The rejections of claims 1, 3-6, 8-11 and 13-15 are the subject of this appeal.

A Notice of Appeal was filed on May 16, 2008.

IV. Status of Amendments

No amendments to the claims have been made.

V. Summary of Claimed Subject Matter

Claim 1 sets forth a computer implemented method for expediting a selected operation in a computer system, comprising associating a plurality of routing operations with an operating system routing task, the plurality of routing operations including the selected operation, wherein the operating system routing task is one of a plurality of operating system tasks executed by an operating system included in the computer system.

See reference number 104 in Fig. 1, and page 5 lines 4-5 in the Specification. See also reference number 102 in Fig. 1, and lines 27-28 on page 4.

Claim 1 further comprises executing the operating system routing task at a low priority level prior to performing the selected operation. See reference numbers 202 and 204 in Fig. 2, and reference numbers 302 and 304 in Fig. 3. See also lines 15-16 on page 5.

Claim 1 also comprises raising the operating system routing task to a high priority level in order to perform the selected operation in response to a detection of a trigger condition comprising a link state advertisement protocol message indicating that the selected operation is to be performed, wherein the raising the operating system routing task to the high priority level causes the operating system routing task to execute without being interrupted by at least one other operating system task running at the low priority. See reference number 210 in Fig. 2, and reference number 310 in Fig. 3. See also lines 17-18 on page 5, line 5 on page 6, and lines 15-16 on page 2.

Claim 3 sets forth that the link state advertisement protocol message includes link status information. See lines 16-17 on page 4.

Claim 4 sets forth that the selected operation is a Dijkstra shortest path computation utilizing the link status information received in the link state advertisement protocol message. See page 4, lines 17-19.

Claim 5 sets forth lowering the operating system routing task to the low priority level upon completion of the selected operation. See reference number 216 in Fig. 2 and reference number 316 in Fig. 3, lines 22-23 on page 5, and lines 8-9 on page 6.

Claim 6 sets forth a computer device (see reference number 100 in Fig. 1) comprising an operating system stored on a computer readable medium, the operating system comprising an operating system task including logic which when executed performs a plurality of routing operations, the plurality of routing operations including a selected operation, wherein the operating system task is one of a plurality of operating system tasks executed by an operating system included in the computer system. See reference number 102 in Fig. 1, and lines 4-8 on page 3.

Further in claim 6, the operating system includes task priority control logic (see reference number 108 in Fig. 1, reference number 200 in Fig. 2, and reference number

300 in Fig. 3) operably coupled to execute the operating system task at a low priority level prior to performing the selected operation (see reference numbers 202 and 204 in Fig. 2, and reference numbers 302 and 304 in Fig. 3, and also lines 15-16 on page 5) and raise the operating system task to a high priority level in order to perform the selected operation upon detection of a trigger condition, the trigger condition comprising receipt of a link state advertisement protocol message (see reference number 210 in Fig. 2, and reference number 310 in Fig. 3. See also lines 17-18 on page 5, and line 5 on page 6), wherein the raising the operating system task to the high priority level causes the operating system task to execute without being interrupted by at least one other operating system task running at the low priority (see lines 15-16 on page 2).

Claim 8 sets forth that operating system task is a routing task, and wherein the link state advertisement protocol message includes link status information. See lines 16-17 on page 4.

Claim 9 sets forth that the selected operation is a Dijkstra shortest path computation utilizing the link status information received in the link state advertisement protocol message. See page 4, lines 17-19.

Claim 10 sets forth that the task priority control logic is operably coupled to lower the operating system task to the low priority level upon completion of the selected operation. See reference number 216 in Fig. 2 and reference number 316 in Fig. 3, lines 22-23 on page 5, and lines 8-9 on page 6.

Claim 11 sets forth a program product comprising a computer readable medium having embodied therein a computer program for expediting a selected operation in a computer system, the computer program comprising task priority control logic (see reference number 108 in Fig. 1, reference number 200 in Fig. 2, and reference number 300 in Fig. 3) programmed to execute an operating system task associated with a plurality of operations including the selected operation at a low priority level prior to performing the selected operation (see reference numbers 202 and 204 in Fig. 2, and reference numbers 302 and 304 in Fig. 3, and also lines 15-16 on page 5) and raise the operating system task to a high priority level in order to perform the selected operation upon detection of a trigger condition including receipt of a link state advertisement protocol message (see reference number 210 in Fig. 2, and reference number 310 in Fig. 3. See

also lines 17-18 on page 5, and line 5 on page 6), wherein the operating system task is one of a plurality of operating system tasks executed by an operating system included in the computer system, and wherein the raising the operating system routing task to the high priority level causes the operating system routing task to execute without being interrupted by at least one other of the plurality of operating system tasks running at the low priority (see lines 15-16 on page 2).

Claim 13 sets forth that the operating system task is a routing task, and wherein the link state advertisement protocol message includes link status information. See lines 16-17 on page 4.

Claim 14 sets forth that the operation is a Dijkstra shortest path computation utilizing the link status information received in the link state advertisement protocol message. See page 4, lines 17-19.

Claim 15 sets forth that the task priority control logic is programmed to lower the operating system task to the low priority level upon completion of the selected operation. See reference number 216 in Fig. 2 and reference number 316 in Fig. 3, lines 22-23 on page 5, and lines 8-9 on page 6.

VI. Grounds of Rejection to be Reviewed on Appeal

Claims 1, 3-6, 8-11 and 13-15 stand rejected as obvious under 35 U.S.C. 103(a) over the combination of United States patent number 6,148,322 of Sand et al. ("Sand") in view of the Examiner's assertion of "Applicant's Admitted Prior Art" ("AAPA").

VII. Argument

The Examiner has failed to establish a *prima facie* case of obviousness under 35 U.S.C. §103(a) in the rejection of claims 1, 3-6, 8-11 and 13-15 using the combination of Sand and AAPA.

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). Appellant asserts that the combination of Sand and AAPA fails to teach or suggest the limitations of the present independent claims 1, 6 and 11, which each include raising an operating system routing task to a high priority level in order to perform a selected operation in response to a detection of a trigger condition comprising a link state advertisement protocol message indicating that the selected operation is to be performed.

If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Appellant further asserts that the dependent claims 3-5, 8-10, and 13-15 are also nonobvious over the combination of Sand and AAPA based on the nonobviousness of independent claims 1, 6 and 11.

United States patent number 6,148,322 of Sand et al. ("Sand")

Sand discloses a processing unit with an improved ability to coordinate the execution of multiple tasks with varying priorities. Tasks to be executed in Sand are assigned both a request condition and a terminating condition, with the processing unit initiating execution of the task with the highest priority whose

request condition is satisfied. In general, the processing unit in Sand terminates an executing task once the terminating condition of that task is satisfied, and then initiates execution of the next highest-priority task with a satisfied request condition. See Abstract.

Sand further discloses that tasks can be associated with priorities. Specifically, Sand teaches that efficient processing by the central processing unit is facilitated by dividing individual jobs to be processed into a set of separate tasks, each of which can be executed by the central processing unit . Each task in Sand is then assigned a respective priority. See lines 41-48 in column 3 of Sand. Examples of the tasks for an embodiment of the Sand system are illustrated in Figure 2 of Sand, with the lowest priority task at the bottom of the list and the highest priority task at the top.

In column 4, lines 15-30 Sand teaches that the microprocessor of the central processing unit in the Sand system is capable of executing only one task at any given instant. Thus, every task in Sand (except for the cyclically-repeated task "zwT") is assigned a request condition and a terminating condition to assist in coordinating execution of the various tasks. During operation of the Sand system, if a request condition for a task having a priority greater than that of the currently-executing task becomes satisfied, the central processing unit will terminate the currently-executing task and initiate execution of the higher priority task. Sand expressly sets forth that the execution of the newly-initiated task will not be terminated until either (a) the request condition for a task having an even higher priority is satisfied; or (b) the terminating condition of the task becomes

satisfied, and that the central processing unit in this way executes tasks in an event-driven manner.

Sand describes the task priorities of the Sand system as follows in lines 7-12 of column 1:

> The present invention relates to processing units for computer systems, and in particular to a processing unit capable of coordinating the execution of multiple tasks having varying degrees of priority. Such processing units typically include a high-priority task, a low-priority task that is always executable, and numerous additional tasks having priorities at varying levels between the high-priority and low-priority tasks. Each of these tasks is executable by the processing unit, but only one task may be executed at any given point in time. Accordingly, the processing unit must coordinate the execution of these tasks to ensure all such tasks are executed in timely fashion.

Thus Sand describes a system in which individual tasks are associated with static, unchanged priorities, and neither describes nor suggests changing the priority associated with any task.

"Applicant's Admitted Prior Art" ("AAPA")

On page 1 of Applicant's Specification, in lines 19-31, in the Background section, there is a general discussion of "link state" routing protocols that may be used to determine routes based upon the status of communication links between nodes. Link State Advertisements ("LSA"s) are mentioned, as well as the use of such messages in previous systems to update topology databases. The Dijkstra shortest path algorithm is also described as one technique for computing shortest paths to all destinations from a single source.

Claims 1, 3-6, 8-11, and 13-15:

Independent claim 1 includes the limitation of "raising the operating system routing task to a high priority level in order to perform the selected operation in response to detection of a trigger condition comprising a link state advertisement protocol message indicating that the selected operation is to be performed." Independent claims 6 and 11 include analogous limitations. This feature of the presently claimed invention is not disclosed or suggested by the combination of Sand and AAPA.

In the Final Office Action, on page 3, the Examiner asserts the presently claim limitation of "raising the operating routing system task to a high priority" is taught at column 2, lines 12-16 of the Summary of the Invention in Sand. Applicant respectfully disagrees with this analysis of Sand. In clear contradistinction, a careful reading of the entirety of Sand, including column 2 and the other above cited sections of Sand, reveals that Sand teaches that execution of a different task having a higher priority is initiated when the request condition is satisfied for the higher priority task. Applicant specifically notes the use of the term "abort" with regard to the currently executing task, and the phrase "initiate execution" with regard to the higher priority task in the text of Sand from lines 12-16 in column 2:

> . . . In one embodiment of the present invention, a processing unit will *abort* a currently-executing task upon satisfaction of the request condition

> for a task having a higher priority, and will then *initiate execution* of the higher priority task. . . (emphasis added)

The above section of Sand thus expressly teaches that the lower priority task is first *aborted*, and then the higher priority task is subsequently *initiated*. The lower priority and higher priority tasks of Sand are clearly different tasks, whose execution does not overlap. Moreover, in the sentence that follows, Sand again confirms that a task in the Sand system has only one priority during its entire execution:

> Processing of the higher-priority task continues until a predetermined maximum processing time elapses, at which time the task is aborted in favor of a task having a next highest priority for which the request condition is satisfied and the terminating condition is not satisfied.

Thus the only way to change from a first task having one priority to a second task having a different priority in Sand is to abort the first task and initiate the execution of the second task. Sand includes no hint or suggestion of even the desirability of changing the priority of a given task in any way.

Combining the teachings of AAPA regarding the previous existence of Link State Advertisements ("LSA"s) and/or the Dijkstra shortest path algorithm with Sand fails to remedy the shortcomings in the Examiner's rejection with regard to the teachings of Sand since like Sand, AAPA also includes no teaching or suggestion of "raising the operating system routing task to a high priority level in order to perform the selected operation in response to detection of a trigger condition comprising a link state advertisement protocol message indicating that the selected operation is to be performed," as in the present independent claims.

For these reasons Applicant respectfully urges that the combination of Sand and AAPA does not describe all the features of the present independent claims 1, 6 and 11. Accordingly, the combination of Sand and AAPA does not support a *prima facie* case of obviousness with regard to the present independent claims 1, 6 and 11 under 35 U.S.C. 103. As to claims 3-5, 8-10 and 13-15, they each depend from independent claims 1, 6 and 11, and are believed to be patentable over the combination of Sand and AAPA for at least the same reasons.

Conclusion

For the reasons above, Appellant respectfully submits that the rejections of the present claims under 35 U.S.C. 103 are improper for at least the reasons set forth above. Appellant accordingly requests that the rejections be withdrawn and the pending claims be allowed.

Respectfully submitted,

NORTEL NETWORKS LIMITED

By: /David Dagg/
David A. Dagg
Reg. No. 37,809
Attorney for Assignee

Date: September 15, 2008

David A. Dagg – Patent Attorney, P.C.
44 Chapin Road
Newton MA 02459
(617) 630-1131

VIII. Claims Appendix

1. (previously presented) A computer implemented method for expediting a selected operation in a computer system, the method comprising:

associating a plurality of routing operations with an operating system routing task, the plurality of routing operations including the selected operation, wherein the operating system routing task is one of a plurality of operating system tasks executed by an operating system included in the computer system;

executing the operating system routing task at a low priority level prior to performing the selected operation; and

raising the operating system routing task to a high priority level in order to perform the selected operation in response to a detection of a trigger condition comprising a link state advertisement protocol message indicating that the selected operation is to be performed, wherein the raising the operating system routing task to the high priority level causes the operating system routing task to execute without being interrupted by at least one other operating system task running at the low priority.

2. (cancelled)

3. (previously presented) The computer implemented method of claim 1, wherein the link state advertisement protocol message includes link status information.

4. (previously presented) The computer implemented method of claim 3, wherein the selected operation is a Dijkstra shortest path computation utilizing the link status information received in the link state advertisement protocol message.

5. (previously presented) The computer implemented method of claim 1, further comprising:

lowering the operating system routing task to the low priority level upon completion of the selected operation.

6. (previously presented) A computer device comprising:

an operating system stored on a computer readable medium, the operating system comprising:

an operating system task including logic which when executed performs a plurality of routing operations, the plurality of routing operations including a selected operation, wherein the operating system task is one of a plurality of operating system tasks executed by an operating system included in the computer system; and

the operating system including task priority control logic operably coupled to execute the operating system task at a low priority level prior to performing the selected operation and raise the operating system task to a high priority level in order to perform the selected operation upon detection of a trigger condition, the trigger condition comprising receipt of a link state advertisement protocol message, wherein the raising the operating system task to the high priority level causes the operating system task to execute without being interrupted by at least one other operating system task running at the low priority.

7. (cancelled)

8. (previously presented) A computer device of claim 6, wherein the operating system task is a routing task, and wherein the link state advertisement protocol message includes link status information.

9. (previously presented) The computer device of claim 8, wherein the selected operation is a Dijkstra shortest path computation utilizing the link status information received in the link state advertisement protocol message.

10. (previously presented) The computer device of claim 6, wherein the task priority control logic is operably coupled to lower the operating system task to the low priority level upon completion of the selected operation.

11. (previously presented) A program product comprising a computer readable medium having embodied therein a computer program for expediting a selected operation in a computer system, the computer program comprising:

task priority control logic programmed to execute an operating system task associated with a plurality of operations including the selected operation at a low priority level prior to performing the selected operation and raise the operating system task to a high priority level in order to perform the selected operation upon detection of a trigger condition including receipt of a link state advertisement protocol message, wherein the operating system task is one of a plurality of operating system tasks executed by an operating system included in the computer system, and wherein the raising the operating

system routing task to the high priority level causes the operating system routing task to execute without being interrupted by at least one other of the plurality of operating system tasks running at the low priority.

12. (cancelled)

13. (previously presented) The program product of claim 11, wherein the operating system task is a routing task, and wherein the link state advertisement protocol message includes link status information.

14. (original) The program product of claim 13, wherein the operation is a Dijkstra shortest path computation utilizing the link status information received in the link state advertisement protocol message.

15. (original) The program product of claim 11, wherein the task priority control logic is programmed to lower the operating system task to the low priority level upon completion of the selected operation.

IX. Evidence Appendix

None.

X. Related Proceedings Appendix

None.